Approximate Nearest Neighbor Libraries

Created	@March 24, 2020 11:40 AM
Updated	@April 13, 2022 4:00 PM
Status	In Production

Started at: @March 13, 2020

Contributors: List contributors

NMSLIB

https://github.com/nmslib/nmslib

https://github.com/nmslib/nmslib/blob/master/python_bindings/parameters.md

Pros

- It’s over 10x faster than Annoy
- It is possible to query using vectors of different dimensions than the one which were indexed.
- Batch query and batch indexing is possible.
- Supports a lot of similarity measures.
- Uses a graph based approach which are current state of the art for ANN.
- Easy install with pip.
- Most accurate at the moment. Check this blog.

Cons

- Not under active development. Last commit was almost 6 months ago.
- Lots of tunable parameters for optimizing the performance according to our needs.

Aa Title	# Average Indexing Time (sec)Indexed 10^5 vectors of dimensionality=1000	# Average Query Time (sec)Queried 1 vector.
Untitled	263.991	0.0013

Hnswlib

https://github.com/nmslib/hnswlib

https://github.com/jlmelville/rcpphnsw

It's a smaller independent sub-project of NMSLIB

Pros

- Light weight, not many dependencies
- It is possible to query using vectors of different dimensions than the one which were indexed.
- Batch query and batch indexing is possible.
- Uses a graph based approach which are current state of the art for ANN.

Cons

- Only supports l2, cosine and inner similarity measures.
- Not many stars on github which indicates not many people are using it.
- Default parameters giving poor approximate nearest neighbours. When tried to query an already indexed element, the nearest neighbour was not the element itself.
- Lots of tunable parameters for optimizing the performance according to our needs.

Aa Title	# Average Indexing Time (sec)Indexed 10^5 vectors of dimensionality=1000	# Average Query Time (sec)Queried 1 vector.

Aa Title	# Average Indexing Time (sec)Indexed 10^5 vectors of dimensionality=1000	# Average Query Time (sec)Queried 1 vector.
Untitled	113.449	0.0014

Faiss

https://github.com/facebookresearch/faiss

Pros

- Has over 6k stars on github.
- Has support for both cpu and gpu implementations.
- Under active development, last commit was made 10 days ago.
- Good documentation.
- Very low indexing time.
- Incremental index update is possible. We may need to retrain the index if we believe that the new data might disturb the existing distribution a lot. But this shouldn’t be a problem since the total indexing time is less than 2 sec.
- Uses a graph based approach which are current state of the art for ANN.
- Easy to use and install.

Cons

- Slow query time with default parameters. (.04 sec)
- It is not possible to query using vectors of different dimensions than the one which were indexed.
- Lots of tunable parameters for optimizing the performance according to our needs.
- Uses some kind of cell quantization techniques to speed up the query time for HNSW. Without this quantization query response times are almost 10x slower than NMSLIB.
- Available similarity metric are - l2 and dot product.
- Slightly less accurate than NMSLIB. Check this blog.

Aa Title	# Average Indexing Time (sec)Indexed 10^5 vectors of dimensionality=1000	# Average Query Time (sec)Queried 1 vector.
Untitled	3.005	0.009

FLANN

https://www.cs.ubc.ca/research/flann/

https://github.com/primetang/pyflann

Pros

Cons

- Pretty old lib.
- Very poor documentation.
- Python binding seemed a bit buggy (not compatible with python3) and not that straightforward to use. I tried to fix some issues but couldn't make it work. Didn't spend much time looking into it.

SPTAG

https://blogs.microsoft.com/ai/bing-vector-search/

https://github.com/microsoft/SPTAG/blob/master/docs/GettingStart.md

Pros

- It supports online updation of indices.

Cons

- Not that straightforward to install.
- Takes forever to build indices for 1000 dimensional 10^5 vectors. I doubt if Microsoft released the entire code because it shouldn't be this slow.
- Very new, so might not be mature enough compared to other libraries.
- No documentation at all.

Aa Title	# Average Indexing Time (sec)Indexed 10^5 vectors of dimensionality=10	# Average Query Time (sec)Queried 1 vector.

Aa Title	# Average Indexing Time (sec)Indexed 10^5 vectors of dimensionality=10	# Average Query Time (sec)Queried 1 vector.
Untitled	956.347	0.0002

Aa Title	# Average Indexing Time (sec)Indexed 10^5 vectors of dimensionality=1000	# Average Query Time (sec)Queried 1 vector.
Untitled	9392.894	0.032